Attorney Docket No.: 16869B-082700US
Client ref. No.: HAL 279

PATENT APPLICATION

METHOD AND APPARATUS FOR BACKUP AND RECOVERY USING STORAGE BASED JOURNALING

Inventor: Kenji Yamagami, a citizen of Japan residing at
108 Calle Nivel,
Los Gatos, CA 95032

Assignee: HITACHI, LTD.
6, Kanda Surugadai 4-chome
Chiyoda-ku
Tokyo 101-8010, Japan
Incorporation: Japan

Entity: Large

TOWNSEND and TOWNSEND and CREW LLP
Two Embarcadero Center, 8th Floor
San Francisco, California 94111-3834
Tel: 650-326-2400

METHOD AND APPARATUS FOR BACKUP AND RECOVERY USING STORAGE BASED JOURNALING

CROSS-REFERENCES TO RELATED APPLICATIONS

[01] This application is related to the following commonly owned and co-pending U.S. applications:

"Method and Apparatus for Data Recovery Using Storage Based Journaling," Attorney Docket Number 16869B-082800US, and

"Method and Apparatus for Synchronizing Applications for Data Recovery Using Storage Based Journaling," Attorney Docket Number 16869B-082900US,

both of which are herein incorporated by reference for all purposes.

BACKGROUND OF THE INVENTION

[02] The present invention is related to computer storage and in particular to backup and recovery of data.

[03] Several methods are conventionally used to prevent the loss of data. Typically, data is backed up in a periodic manner (e.g., once a day) by a system administrator. Many systems are commercially available which provide backup and recovery of data; e.g., Veritas NetBackup, Legato/Networker, and so on. Another technique is known as volume shadowing. This technique produces a mirror image of data onto a secondary storage system as it is being written to the primary storage system.

[04] Journaling is a backup and restore technique commonly used in database systems. An image of the data to be backed up is taken. Then, as changes are made to the data, a journal of the changes is maintained. Recovery of data is accomplished by applying the journal to an appropriate image to recover data at any point in time. Typical database systems, such as Oracle, can perform journaling.

[05] Except for database systems, however, there are no ways to recover data at any point in time. Even for database systems, applying a journal takes time since the procedure includes:

- reading the journal data from storage (e.g., disk)
- the journal must be analyzed to determine at where in the journal the desired data can be found

- apply the journal data to a suitable image of the data to reproduce the activities performed on the data - this usually involves accessing the image, and writing out data as the journal is applied

[06] Recovering data at any point in time addresses the following types of administrative requirements. For example, a typical request might be, "I deleted a file by mistake at around 10:00 am yesterday. I have to recover the file just before it was deleted."

[07] If the data is not in a database system, this kind of request cannot be conveniently, if at all, serviced. A need therefore exists for processing data in a manner that facilitates recovery of lost data. A need exists for being able to provide data processing that facilitates data recovery in user environments other than in a database application.

SUMMARY OF THE INVENTION

[08] A storage system provides data storage services for users and their applications. The storage system performs additional data processing to provide for recovery of lost data, including performing snapshot operations and journaling. Snapshots and journal entries are stored separately from the production data volumes provided for the users. Older journal entries are cleared in order to make for new journal entries. This involves updating a snapshot by applying one or more of the older journal entries to an appropriate snapshot. Subsequent recovery of lost data can be provided by accessing an appropriate snapshot and applying journal entries to the snapshot to reproduce the desired data state.

BRIEF DESCRIPTION OF THE DRAWINGS

[09] Aspects, advantages and novel features of the present invention will become apparent from the following description of the invention presented in conjunction with the accompanying drawings:

Fig. 1 is a high level generalized block diagram of an illustrative embodiment of the present invention;

Fig. 2 is a generalized illustration of a illustrative embodiment of a data structure for storing journal entries in accordance with the present invention;

Fig. 3 is a generalized illustration of an illustrative embodiment of a data structure for managing the snapshot volumes and the journal entry volumes in accordance with the present invention;

Fig. 4 is a high level flow diagram highlighting the processing between the recovery manager and the controller in the storage system;

Fig. 5 illustrates the relationship between a snapshot and a plurality of journal entries;

Fig. 5A illustrates the relationship among a plurality of snapshots and a plurality of journal entries;

Fig. 6 is a high level illustration of the data flow when an overflow condition arises;

Fig. 7 is a high level flow chart highlighting an aspect of the controller in the storage system to handle an overflow condition; and

Fig. 7A illustrates an alternative to a processing step shown in Fig. 7.

DESCRIPTION OF THE SPECIFIC EMBODIMENTS

[10] Fig. 1 is a high level generalized block diagram of an illustrative embodiment of a backup and recovery system according to the present invention. When the system is activated, a snapshot is taken for production data volumes (DVOL) 101. The term "snapshot" in this context conventionally refers to a data image of at the data volume at a given point in time. Depending on system requirements, implementation, and so on, the snapshot can be of the entire data volume, or some portion or portions of the data volume(s). During the normal course of operation of the system in accordance with the invention, a journal entry is made for every write operation issued from the host to the data volumes. As will be discussed below, by applying a series of journal entries to an appropriate snapshot, data can be recovered at any point in time.

[11] The backup and recovery system shown in Fig. 1 includes at least one storage system 100. Though not shown, one of ordinary skill can appreciate that the storage system includes suitable processor(s), memory, and control circuitry to perform IO between a host 110 and its storage media (e.g., disks). The backup and recovery system also requires at least one host 110. A suitable communication path 130 is provided between the host and the storage system.

[12] The host 110 typically will have one or more user applications (APP) 112 executing on it. These applications will read and/or write data to storage media contained in the data volumes 101 of storage system 100. Thus, applications 112 and the data volumes 101 represent the target resources to be protected. It can be appreciated that data used by the user applications can be stored in one or more data volumes.

[13] In accordance with the invention, a journal group (JNLG) 102 is defined. The data volumes 101 are organized into the journal group. In accordance with the present invention,

a journal group is the smallest unit of data volumes where journaling of the write operations from the host 110 to the data volumes is guaranteed. The associated journal records the order of write operations from the host to the data volumes in proper sequence. The journal data produced by the journaling activity can be stored in one or more journal volumes(JVOL) 106.

[14] The host 110 also includes a recovery manager (RM) 111. This component provides a high level coordination of the backup and recovery operations. Additional discussion about the recovery manager will be discussed below.

[15] The storage system 100 provides a snapshot (SS) 105 of the data volumes comprising a journal group. For example, the snapshot 105 is representative of the data volumes 101 in the journal group 106 at the point in time that the snapshot was taken. Conventional methods are known for producing the snapshot image. One or more snapshot volumes (SVOL) 107 are provided in the storage system which contain the snapshot data. A snapshot can be contained in one or more snapshot volumes. Though the disclosed embodiment illustrates separate storage components for the journal data and the snapshot data, it can be appreciated that other implementations can provide a single storage component for storing the journal data and the snapshot data.

[16] A management table (MT) 108 is provided to store the information relating to the journal group 102, the snapshot 105, and the journal volume(s) 106. Fig. 3 and the accompanying discussion below reveal additional detail about the management table.

[17] A controller component 140 is also provided which coordinates the journaling of write operations and snapshots of the data volumes, and the corresponding movement of data among the different storage components 101, 106, 107. It can be appreciated that the controller component is a logical representation of a physical implementation which may comprise one or more sub-components distributed within the storage system 100.

[18] Fig. 2 shows the data used in an implementation of the journal. When a write request from the host 110 arrives at the storage system 100, a journal is generated in response. The journal comprises a Journal Header 219 and Journal Data 225. The Journal Header 219 contains information about its corresponding Journal Data 225. The Journal Data 225 comprises the data (write data) that is the subject of the write operation. This kind of journal is also referred to as an "AFTER journal."

[19] The Journal Header 219 comprises an offset number (JH_OFS) 211. The offset number identifies a particular data volume 101 in the journal group 102. In this particular implementation, the data volumes are ordered as the 0^{th} data volume, the 1^{st} data volume, the 2^{nd} data volume and so on. The offset numbers might be 0, 1, 2, etc.

[20] A starting address in the data volume (identified by the offset number 211) to which the write data is to be written is stored to a field in the Journal Header 219 to contain an address (JH_ADR) 212. For example, the address can be represented as a block number (LBA, Logical Block Address).

[21] A field in the Journal Header 219 stores a data length (JH_LEN) 213, which represents the data length of the write data. Typically it is represented as a number of blocks.

[22] A field in the Journal Header 219 stores the write time (JH_TIME) 214, which represents the time when the write request arrives at the storage system 100. The write time can include the calendar date, hours, minutes, seconds and even milliseconds. This time can be provided by the disk controller 140 or by the host 110. For example, in a mainframe computing environment, two or more mainframe hosts share a timer and can provide the time when a write command is issued.

[23] A sequence number(JH_SEQ) 215 is assigned to each write request. The sequence number is stored in a field in the Journal Header 219. Every sequence number within a given journal group 102 is unique. The sequence number is assigned to a journal entry when it is created.

[24] A journal volume identifier (JH_JVOL) 216 is also stored in the Journal Header 219. The volume identifier identifies the journal volume 106 associated with the Journal Data 225. The identifier is indicative of the journal volume containing the Journal Data. It is noted that the Journal Data can be stored in a journal volume that is different from the journal volume which contains the Journal Header.

[25] A journal data address (JH_JADR) 217 stored in the Journal Header 219 contains the beginning address of the Journal Data 225 in the associated journal volume 106 that contains the Journal Data.

[26] Fig. 2 shows that the journal volume 106 comprises two data areas: a Journal Header Area 210 and a Journal Data Area 220. The Journal Header Area 210 contains only Journal Headers 219, and Journal Data Area 220 contains only Journal Data 225. The Journal Header is a fixed size data structure. A Journal Header is allocated sequentially from the beginning of the Journal Header Area. This sequential organization corresponds to the chronological order of the journal entries. As will be discussed, data is provided that points to the first journal entry in the list, which represents the "oldest" journal entry. It is typically necessary to find the Journal Header 219 for a given sequence number (as stored in the sequence number field 215) or for a given write time (as stored in the time field 214).

[27] Journal Header 219 and Journal Data 225 are contained in chronological order in their respective areas in the journal volume 106. Thus, the order in which the Journal Header and the Journal Data are stored in the journal volume is the same order as the assigned sequence number. As will be discussed below, an aspect of the present invention is that the journal information 219, 225 wrap within their respective areas 210, 220.

[28] Fig. 3 shows detail about the management table 108 (Fig. 1). In order to manage the Journal Header Area 210 and Journal Data Area 220, pointers for each area are needed. As mentioned above, the management table maintains configuration information about a journal group 102 and the relationship between the journal group and its associated journal volume(s) 106 and snapshot image 105.

[29] The management table 300 shown in Fig. 3 illustrates an example management table and its contents. The management table stores a journal group ID (GRID) 310 which identifies a particular journal group 102 in a storage system 100. A journal group name (GRNAME) 311 can also be provided to identify the journal group with a human recognizable identifier.

[30] A journal attribute (GRATTR) 312 is associated with the journal group 102. In accordance with this particular implementation, two attributes are defined: MASTER and RESTORE. The MASTER attribute indicates the journal group is being journaled. The RESTORE attribute indicates that the journal group is being restored from a journal.

[31] A journal status (GRSTS) 315 is associated with the journal group 102. There are two statuses: ACTIVE and INACTIVE.

[32] The management table includes a field to hold a sequence counter (SEQ) 313. This counter serves as the source of sequence numbers used in the Journal Header 219. When creating a new journal, the sequence number 313 is read and assigned to the new journal. Then, the sequence number is incremented and written back into the management table.

[33] The number (NUM_DVOL) 314 of data volumes 101 contained in a give journal group 102 is stored in the management table.

[34] A data volume list (DVOL_LIST) 320 lists the data volumes in a journal group. In a particular implementation, DVOL_LIST is a pointer to the first entry of a data structure which holds the data volume information. This can be seen in Fig. 3. Each data volume information comprises an offset number (DVOL_OFFS) 321. For example, if the journal group 102 comprises three data volumes, the offset values could be 0, 1 and 2. A data volume identifier (DVOL_ID) 322 uniquely identifies a data volume within the entire storage

system 100. A pointer (DVOL_NEXT) 324 points to the data structure holding information for the next data volume in the journal group; it is a NULL value otherwise.

[35] The management table includes a field to store the number of journal volumes (NUM_JVOL) 330 that are being used to contain the data (journal header and journal data) associated with a journal group 102.

[36] As described in Fig. 2, the Journal Header Area 210 contains the Journal Headers 219 for each journal; likewise for the Journal Data components 225. As mentioned above, an aspect of the invention is that the data areas 210, 220 wrap. This allows for journaling to continue despite the fact that there is limited space in each data area.

[37] The management table includes fields to store pointers to different parts of the data areas 210, 220 to facilitate wrapping. Fields are provided to identify where the next journal entry is to be stored. A field (JI_HEAD_VOL) 331 identifies the journal volume 106 that contains the Journal Header Area 210 which will store the next new Journal Header 219. A field (JI_HEAD_ADR) 332 identifies an address on the journal volume of the location in the Journal Header Area where the next Journal Header will be stored. The journal volume that contains the Journal Data Area 220 into which the journal data will be stored is identified by information in a field (JI_DATA_VOL) 335. A field (JI_DATA_ADR) 336 identifies the specific address in the Journal Data Area where the data will be stored. Thus, the next journal entry to be written is "pointed" to by the information contained in the "JI_" fields 331, 332, 335, 336.

[38] The management table also includes fields which identify the "oldest" journal entry. The use of this information will be described below. A field (JO_HEAD_VOL) 333 identifies the journal volume which stores the Journal Header Area 210 that contains the oldest Journal Header 219. A field (JO_HEAD_ADR) 334 identifies the address within the Journal Header Area of the location of the journal header of the oldest journal. A field (JO_DATA_VOL) 337 identifies the journal volume which stores the Journal Data Area 220 that contains the data of the oldest journal. The location of the data in the Journal Data Area is stored in a field (JO_DATA_ADR) 338.

[39] The management table includes a list of journal volumes (JVOL_LIST) 340 associated with a particular journal group 102. In a particular implementation, JVOL_LIST is a pointer to a data structure of information for journal volumes. As can be seen in Fig. 3, each data structure comprises an offset number (JVOL_OFS) 341 which identifies a particular journal volume 106 associated with a given journal group 102. For example, if a journal group is associated with two journal volumes 106, then each journal volume might be

identified by a 0 or a 1. A journal volume identifier (JVOL_ID) 342 uniquely identifies the journal volume within the storage system 100. Finally, a pointer (JVOL_NEXT) 344 points to the next data structure entry pertaining to the next journal volume associated with the journal group; it is a NULL value otherwise.

[40] The management table includes a list (SS_LIST) 350 of snapshot images 105 associated with a given journal group 102. In this particular implementation, SS_LIST is a pointer to snapshot information data structures, as indicated in Fig. 3. Each snapshot information data structure includes a sequence number (SS_SEQ) 351 that is assigned when the snapshot is taken. As discussed above, the number comes from the sequence counter 313. A time value (SS_TIME) 352 indicates the time when the snapshot was taken. A status (SS_STS) 358 is associated with each snapshot; valid values include VALID and INVALID. A pointer (SS_NEXT) 353 points to the next snapshot information data structure; it is a NULL value otherwise.

[41] Each snapshot information data structure also includes a list of snapshot volumes 107 (Fig. 1) used to store the snapshot images 105. As can be seen in Fig. 3, a pointer (SVOL_LIST) 354 to a snapshot volume information data structure is stored in each snapshot information data structure. Each snapshot volume information data structure includes an offset number (SVOL_OFFS) 355 which identifies a snapshot volume that contains at least a portion of the snapshot image. It is possible that a snapshot image will be segmented or otherwise partitioned and stored in more than one snapshot volume. In this particular implementation, the offset identifies the i^{th} snapshot volume which contains a portion (segment, partition, etc) of the snapshot image. In one implementation, the i^{th} segment of the snapshot image might be stored in the i^{th} snapshot volume. Each snapshot volume information data structure further includes a snapshot volume identifier (SVOL_ID) 356 that uniquely identifies the snapshot volume in the storage system 100. A pointer (SVOL_NEXT) 357 points to the next snapshot volume information data structure for a given snapshot image.

[42] Fig 4 shows a flowchart highlighting the processing performed by the recovery manager 111 and Storage System 100 to initiate backup processing in accordance with the illustrative embodiment of the invention as shown in the figures. If journal entries are not recorded during the taking of a snapshot, the write operations corresponding to those journal entries would be lost and data corruption could occur during a data restoration operation. Thus, in accordance with an aspect of the invention, the journaling process is started prior to taking the first snapshot. Doing this ensures that any write operations which occur during the

taking of a snapshot are journaled. As a note, any journal entries recorded prior to the completion of the snapshot can be ignored.

[43] Further in accordance with the invention, a single sequence of numbers (SEQ) 313 are associated with each of one or more snapshots and journal entries, as they are created. The purpose of associating the same sequence of numbers to both the snapshots and the journal entries will be discussed below.

[44] Continuing with Fig. 4, the recovery manager 111 might define, in a step 410, a journal group (JNLG) 102 if one has not already been defined. As indicated in Fig. 1, this may include identifying one or data volumes (DVOL) 101 for which journaling is performed, and identifying one or journal volumes (JVOL) 106 which are used to store the journal-related information. The recovery manager performs a suitable sequence of interactions with the storage system 100 to accomplish this. In a step 415, the storage system may create a management table 108 (Fig. 1), incorporating the various information shown in the table detail 300 illustrated in Fig. 3. Among other things, the process includes initializing the JVOL_LIST 340 to list the journal volumes which comprise the journal group 102 Likewise, the list of data volumes DVOL_LIST 320 is created. The fields which identify the next journal entry (or in this case where the table is first created, the first journal entry) are initialized. Thus, JI_HEAD_VOL 331 might identify the first in the list of journal volumes and JI_HEAD_ADR 332 might point to the first entry in the Journal Header Area 210 located in the first journal volume. Likewise, JI_DATA_VOL 335 might identify the first in the list of journal volumes and JI_DATA_ADR 336 might point to the beginning of the Journal Data Area 220 in the first journal volume. Note, that the header and the data areas 210, 220 may reside on different journal volumes, so JI_DATA_VOL might identify a journal volume different from the first journal volume.

[45] In a step 420, the recovery manager 111 will initiate the journaling process. Suitable communication(s) are made to the storage system 100 to perform journaling. In a step 425, the storage system will make a journal entry (also referred to as an "AFTER journal") for each write operation that issues from the host 110.

[46] With reference to Fig. 3, making a journal entry includes, among other things, identifying the location for the next journal entry. The fields JI_HEAD_VOL 331 and JI_HEAD_ADR 332 identify the journal volume 106 and the location in the Journal Header Area 210 of the next Journal Header 219. The sequence counter (SEQ) 313 from the management table is copied to (associated with) the JH_SEQ 215 field of the next header. The sequence counter is then incremented and stored back to the management table. Of

course, the sequence counter can be incremented first, copied to JH_SEQ, and then stored back to the management table.

[47] The fields JI_DATA_VOL 335 and in the management table identify the journal volume and the beginning of the Journal Data Area 220 for storing the data associated with the write operation. The JI_DATA_VOL and JI_DATA_ADR fields are copied to JH_JVOL 216 and to JH_ADR 212, respectively, of the Journal Header, thus providing the Journal Header with a pointer to its corresponding Journal Data. The data of the write operation is stored.

[48] The JI_HEAD_VOL 331 and JI_HEAD_ADR 332 fields are updated to point to the next Journal Header 219 for the next journal entry. This involves taking the next contiguous Journal Header entry in the Journal Header Area 210. Likewise, the JI_DATA_ADR field (and perhaps JI_DATA_VOL field) is updated to reflect the beginning of the Journal Data Area for the next journal entry. This involves advancing to the next available location in the Journal Data Area. These fields therefore can be viewed as pointing to a list of journal entries. Journal entries in the list are linked together by virtue of the sequential organization of the Journal Headers 219 in the Journal Header Area 210.

[49] When the end of the Journal Header Area 210 is reached, the Journal Header 219 for the next journal entry wraps to the beginning of the Journal Header Area. Similarly for the Journal Data 225. To prevent overwriting earlier journal entries, the present invention provides for a procedure to free up entries in the journal volume 106. This aspect of the invention is discussed below.

[50] For the very first journal entry, the JO_HEAD_VOL field 333, JO_HEAD_ADR field 334, JO_DATA_VOL field 337, and the JO_DATA_ADR field 338 are set to contain their contents of their corresponding "JI_" fields. As will be explained the "JO_" fields point to the oldest journal entry. Thus, as new journal entries are made, the "JO_" fields do not advance while the "JI_" fields do advance. Update of the "JO_" fields is discussed below.

[51] Continuing with the flowchart of Fig. 4, when the journaling process has been initiated, all write operations issuing from the host are journaled. Then in a step 430, the recovery manager 111 will initiate taking a snapshot of the data volumes 101. The storage system 100 receives an indication from the recovery manager to take a snapshot. In a step 435, the storage system performs the process of taking a snapshot of the data volumes. Among other things, this includes accessing SS_LIST 350 from the management table (Fig. 3). A suitable amount of memory is allocated for fields 351 - 354 to represent the next snapshot. The sequence counter (SEQ) 313 is copied to the field SS_SEQ 351 and

incremented, in the manner discussed above for JH_SEQ 215. Thus, over time, a sequence of numbers is produced from SEQ 313, each number in the sequence being assigned either to a journal entry or a snapshot entry.

[52] The snapshot is stored in one (or more) snapshot volumes (SVOL) 107. A suitable amount of memory is allocated for fields 355 - 357. The information relating to the SVOLs for storing the snapshot are then stored into the fields 355 - 357. If additional volumes are required to store the snapshot, then additional memory is allocated for fields 355 - 357.

[53] Fig. 5 illustrates the relationship between journal entries and snapshots. The snapshot 520 represents the first snapshot image of the data volumes 101 belonging to a journal group 102. Note that journal entries (510) having sequence numbers SEQ0 and SEQ1 have been made, and represent journal entries for two write operations. These entries show that journaling has been initiated at a time prior to the snapshot being taken (step 420). Thus, at a time corresponding to the sequence number SEQ2, the recovery manager 111 initiates the taking of a snapshot, and since journaling has been initiated, any write operations occurring during the taking of the snapshot are journaled. Thus, the write operations 500 associated with the sequence numbers SEQ3 and higher show that those operations are being journaled. As an observation, the journal entries identified by sequence numbers SEQ0 and SEQ1 can be discarded or otherwise ignored.

[54] Recovering data typically requires recover the data state of at least a portion of the data volumes 101 at a specific time. Generally, this is accomplished by applying one or more journal entries to a snapshot that was taken earlier in time relative to the journal entries. In the disclosed illustrative embodiment, the sequence number SEQ 313 is incremented each time it is assigned to a journal entry or to a snapshot. Therefore, it is a simple matter to identify which journal entries can be applied to a selected snapshot; i.e., those journal entries whose associated sequence numbers (JH_SEQ, 215) are greater than the sequence number (SS_SEQ, 351) associated with the selected snapshot.

[55] For example, the administrator may specify some point in time, presumably a time that is earlier than the time (the "target time") at which the data in the data volume was lost or otherwise corrupted. The time field SS_TIME 352 for each snapshot is searched until a time earlier than the target time is found. Next, the Journal Headers 219 in the Journal Header Area 210 is searched, beginning from the "oldest" Journal Header. The oldest Journal Header can be identified by the "JO_" fields 333, 334, 337, and 338 in the management table. The Journal Headers are searched sequentially in the area 210 for the first header whose sequence number JH_SEQ 215 is greater than the sequence number SS_SEQ 351 associated

with the selected snapshot. The selected snapshot is incrementally updated by applying each journal entry, one at a time, to the snapshot in sequential order, thus reproducing the sequence of write operations. This continues as long as the time field JH_TIME 214 of the journal entry is prior to the target time. The update ceases with the first journal entry whose time field 214 is past the target time.

[56] In accordance with one aspect of the invention, a single snapshot is taken. All journal entries subsequent to that snapshot can then be applied to reconstruct the data state at a given time. In accordance with another aspect of the present invention, multiple snapshots can be taken. This is shown in Fig. 5A where multiple snapshots 520' are taken. In accordance with the invention, each snapshot and journal entry is assigned a sequence number in the order in which the object (snapshot or journal entry) is recorded. It can be appreciated that there typically will be many journal entries 510 recorded between each snapshot 520'. Having multiple snapshots allows for quicker recovery time for restoring data. The snapshot closest in time to the target recovery time would be selected. The journal entries made subsequent to the snapshot could then be applied to restore the desired data state.

[57] Fig. 6 illustrates another aspect of the present invention. In accordance with the invention, a journal entry is made for every write operation issued from the host; this can result in a rather large number of journal entries. As time passes and journal entries accumulate, the one or more journal volumes 106 defined by the recovery manager 111 for a journal group 102 will eventually fill up. At that time no more journal entries can be made. As a consequence, subsequent write operations would not be journaled and recovery of the data state subsequent to the time the journal volumes become filled would not be possible.

[58] Fig. 6 shows that the storage system 100 will apply journal entries to a suitable snapshot in response to detection of an "overflow" condition. An "overflow" is deemed to exist when the available space in the journal volume(s) falls below some predetermined threshold. It can be appreciated that many criteria can be used to determine if an overflow condition exists. A straightforward threshold is based on the total storage capacity of the journal volume(s) assigned for a journal group. When the free space becomes some percentage (say, 10%) of the total storage capacity, then an overflow condition exists. Another threshold might be used for each journal volume. In an aspect of the invention, the free space capacity in the journal volume(s) is periodically monitored. Alternatively, the free space can be monitored in an aperiodic manner. For example, the intervals between monitoring can be randomly spaced. As another example, the monitoring intervals can be

spaced apart depending on the level of free space; i.e., the monitoring interval can vary as a function of the free space level.

[59] Fig. 7 highlights the processing which takes place in the storage system 100 to detect an overflow condition. Thus, in a step, 710, the storage system periodically checks the total free space of the journal volume(s) 106; e.g., every ten seconds. The free space can easily be calculated since the pointers (e.g., JI_CTL_VOL 331, JI_CTL_ADDR 332) in the management table 300 maintain the current state of the storage consumed by the journal volumes. If the free space is above the threshold, then the monitoring process simply waits for a period of time to pass and then repeats its check of the journal volume free space.

[60] If the free space falls below a predetermined threshold, then in a step 720 some of the journal entries are applied to a snapshot to update the snapshot. In particular, the oldest journal entry(ies) are applied to the snapshot.

[61] Referring to Fig. 3, the Journal Header 219 of the "oldest" journal entry is identified by the JO_HEAD_VOL field 333 and the JO_HEAD_ADR field 334. These fields identify the journal volume and the location in the journal volume of the Journal Header Area 210 of the oldest journal entry. Likewise, the Journal Data of the oldest journal entry is identified by the JO_DATA_VOL field 337 and the JO_DATA_ADR field 338. The journal entry identified by these fields is applied to a snapshot. The snapshot that is selected is the snapshot having an associated sequence number closest to the sequence number of the journal entry and earlier in time than the journal entry. Thus, in this particular implementation where the sequence number is incremented each time, the snapshot having the sequence number closest to but less than the sequence number of the journal entry is selected (i.e., "earlier in time). When the snapshot is updated by applying the journal entry to it, the applied journal entry is freed. This can simply involve updating the JO_HEAD_VOL field 333, JO_HEAD_ADR field 334, JO_DATA_VOL field 337, and the JO_DATA_ADR field 338 to the next journal entry.

[62] As an observation, it can be appreciated by those of ordinary skill, that the sequence numbers will eventually wrap, and start counting from zero again. It is well within the level of ordinary skill to provide a suitable mechanism for keeping track of this when comparing sequence numbers.

[63] Continuing with Fig. 7, after applying the journal entry to the snapshot to update the snapshot, a check is made of the increase in the journal volume free space as a result of the applied journal entry being freed up (step 730). The free space can be compared against the threshold criterion used in step 710. Alternatively, a different threshold can be used. For

example, here a higher amount of free space may be required to terminate this process than was used to initiate the process. This avoids invoking the process too frequently, but once invoked the second higher threshold encourages recovering as much free space as is reasonable. It can be appreciated that these thresholds can be determined empirically over time by an administrator.

[64] Thus, in step 730, if the threshold for stopping the process is met (i.e., free space exceeds threshold), then the process stops. Otherwise, step 720 is repeated for the next oldest journal entry. Steps 730 and 720 are repeated until the free space level meets the threshold criterion used in step 730.

[65] Fig. 7A highlights sub-steps for an alternative embodiment to step 720 shown in Fig. 7. Step 720 frees up a journal entry by applying it to the latest snapshot that is not later in time than the journal entry. However, where multiple snapshots are available, it may be possible to avoid the time consuming process of applying the journal entry to a snapshot in order to update the snapshot.

[66] Fig. 7A shows details for a step 720' that is an alternate to step 720 of Fig. 7. At a step 721, a determination is made whether a snapshot exists that is later in time than the oldest journal entry. This determination can be made by searching for the first snapshot whose associated sequence number is greater than that of the oldest journal entry. Alternatively, this determination can be made by looking for a snapshot that is a predetermined amount of time later than the oldest journal entry can be selected; for example, the criterion may be that the snapshot must be at least one hour later in time than the oldest journal entry. Still another alternate is to use the sequence numbers associated with the snapshots and the journal entries, rather than time. For example, the criterion might be to select a snapshot whose sequence number is N increments away from the sequence number of the oldest journal entry.

[67] If such a snapshot can be found in step 721, then the earlier journal entries can be removed without having to apply them to a snapshot. Thus, in a step 722, the "JO_" fields (JO_HEAD_VOL 333, JO_HEAD_ADR 334, JO_DATA_VOL 337, and JO_DATA_ADR 338) are simply moved to a point in the list of journal entries that is later in time than the selected snapshot. If no such snapshot can be found, then in a step 723 the oldest journal entry is applied to a snapshot that is earlier in time than the oldest journal entry, as discussed for step 720.

[68] Still another alternative for step 721 is simply to select the most recent snapshot. All the journal entries whose sequence numbers are less than that of the most recent snapshot can

be freed. Again, this simply involves updating the "JO_" fields so they point to the first journal entry whose sequence number is greater than that of the most recent snapshot. Recall that an aspect of the invention is being able to recover the data state for any desired point in time. This can be accomplished by storing as many journal entries as possible and then applying the journal entries to a snapshot to reproduce the write operations. This last embodiment has the potential effect of removing large numbers of journal entries, thus reducing the range of time within which the data state can be recovered. Nevertheless, for a particular configuration it may be desirable to remove large numbers of journal entries for a given operating environment.

[69] It can be appreciated that the foregoing described steps can be embodied entirely in the controller 140 (e.g., a disk controller). This can take on the form of pure software, custom logic, or some suitable combination of software and hardware, depending on the particular implementation. More generally, the foregoing disclosed embodiments typically can be provided using a combination of hardware and software implementations. One of ordinary skill can readily appreciate that the underlying technical solution will be determined based on factors including but not limited or restricted to system cost, system performance, legacy software and legacy hardware, operating environment, and so on. The described current and contemplated embodiments can be readily reduced to specific implementations without undue experimentation by those of ordinary skill in the relevant art.